Manual Verification and Correction of Automatically Labeled Zones: User Interface Considerations

Glenn Pearson[1] **George R. Thoma**[2]
National Library of Medicine (NLM)
Bethesda, Maryland 20894

Abstract

A system for automatic extraction of bibliographic information from scanned document pages is being developed at NLM. A goal of this system is the automated labeling of rectangular image zones as title, authors, abstract, etc. Software to help achieve this goal has two broad roles. The research role contributes to the extraction of salient features from image or OCR data and their subsequent analysis by AI systems being tuned for zone label categorization. The production role embeds such functionality within the planned production workflow. Furthermore, since automated labeling will not be 100% reliable in the near term, an important part of the production role is manual verification and correction of zone segmentation and labeling to achieve full accuracy.

A software-development case study is presented of "Zone Checker", a software component that contributes to both roles. This duality of roles introduces multiple classes of users, with implications for menu structure and other aspects of graphical user interface (GUI) design. Additional design issues are explored, particularly those specific to visualizing, verifying, and correcting zones and their labels. An example is how to best convey the relative reading order among zones for image text that flows from one column to the next.

1 Introduction to MARS, the Medical Article Record System

1.1 Keeping MEDLINE Up to Date

One of the major accomplishments and services of the National Library of Medicine (NLM) is the creation and maintenance of MEDLINE, a database of journal article citations and abstracts covering much of the world's biomedical and related scientific literature. This resource, originally accessible by researchers, doctors and other providers, and medical librarians, has been available since 1998 to anyone with web access[3]. MEDLINE's data is also incorporated into a number of public and private value-added databases.

The traditional method of data entry into MEDLINE, as into most bibliographic databases worldwide requiring high accuracy, is by typing in all information in duplicate. More recently, two additional methods have been implemented that are less labor-intensive:

- Direct electronic submission from publishers; and
- Scanner-based journal imaging, with optical character recognition (OCR). This is the MARS approach.

The portion of journals handled by each of these two new methods continues to grow.

1.2 The Origins of MARS I

MARS I was developed, starting in late 1996, by NLM's Communications Engineering Branch (CEB) in collaboration with other groups within the library. Since installation, MARS I has seen substantial incremental improvements in both software and operations. Throughout 1998, a sustained throughput of over 600 journal articles per weekday was achieved, one third of MEDLINE's total data entry requirements.

1.3 The Origins of MARS II

In parallel with improvements to MARS I, CEB began research on and development of a successor system with three major goals:

- Diminution of manual work per article, by the incorporation of both document image understanding techniques and improved user interfaces;
- Replacement of the intricate MARS I file-based mechanisms with a database system, to provide better reliability, data integrity, and potential for throughput growth;

---

[1] Contact: http://archive.nlm.nih.gov/staff/pearson.php Dr. Pearson is a computer scientist with Management Systems Designers, Inc., a provider of on-site software development services.

[2] http://archive.nlm.nih.gov/staff/thoma.php

[3] See NLM's homepage, www.nlm.nih.gov, for no-cost MEDLINE access via PubMed and Internet Grateful Med.

- Replacement of DOS and Windows 3.1 16-bit client applications with Windows 32-bit ones. Associated with this was a migration from C to C++, along with selective incorporation of OCX componentware.

Since the start of 1997, MARS II has been a major CEB project. At the outset, the project's software developers, generally adept in C, were more heterogeneous with respect to experience with C++ and the DevStudio/ Microsoft Foundation Classes (MFC) environment.[4] Nevertheless, by fall of 1998, a first prototype of client stations working with the database was tested. First production deployment should occur this year.

2 Zone Checker As First Conceived

2.1 Beyond MARS I

In MARS I, the first page of each appropriate journal article was scanned. Next, within each bi-tonal (black and white) image on screen, a few fields were manually "zoned". For each, a rectangle was drawn on the bitmap image by positioning two opposite corners, and what we will call here a "zone label type" (specifically, either Title or Abstract) implicitly assigned by entry order. Only the areas within these zones were OCR'd. Separately, all fields except the abstract entered by typing. The title was entered both ways in order to allow the two sources to be matched and merged.

In MARS II, as part of the goal to minimize or eliminate manual steps, manual prezoning would no longer be done at the scan station. Instead, scanning would be followed by full-page image segmentation (to locate paragraphs) and extensive auto-labeling. In order for the latter step to become highly reliable, multiple sources of information (or analysis techniques or rules) would need to be consulted, some of which are general and others of which are journal-specific. The latter, in aggregate, represent a "journal profile".

[4] C++ was chosen over Java because of the easier skill migration, more mature development tools, and faster run-time performance. Also, Java's strength is in creating distributed remote systems, but MARS requires the operators to be in physical proximity in order to circulate journals, since relying upon bitmap page images alone is sometimes inadequate for character-level inspection. At least this is true for the 300 dpi images that give us the best OCR results; others [10] have also found 300 superior to 400 and 600 dpi with commercial OCR systems. As for choosing MFC, the only real Windows-centric alternative, Borland's OWL, was no longer seen as competitive for new projects not requiring Windows 3.x support.

2.2 The Research Role – Capturing Journal Profiles

Early in the MARS II research program, a need was identified for the development of a software tool to help capture certain components applicable to a journal profile. Such information, once captured, would be analyzed and generalized to form profiles. This tool would read TIFF images generated by the MARS I project. It would perform these steps:

1. Open a scanned image;
2. Algorithmically locate zones. This boundary-detection process could be informed by image segmentation, artificial intelligence (AI), historic journal-specific information about layout and font styles, and, with some limited OCR-like capability, and keyword recognition;
3. Display the zoned image, along with a table of zone label types;
4. Allow the operator to manually assign a label type (e.g., author, title, etc.) to each zone;
5. Store the resulting data (journal, image number, zone sequence number and position, label type) in tabular form as part of a journal profile.

Step 4's verification establishes the ground truth, an essential element when building journal profiles either manually or using AI techniques requiring feedback or learning (e.g., weight adjustments to an expert system or neural net). It also allows quantified assessment of the quality of automatic zone finding.

2.3 The Possible Production Role – Using Journal Profiles

Contemporaneously and independently, an early database design identified certain client modules in the MARS II workflow [Table 1]. This design didn't yet clearly incorporate automated labeling. To do so, auto-labeling must happen no earlier than "Segmentation" and no later than "Zone Validation".

Consider the three validation stages shown. The last two of these, OCR and record validation, are handled in MARS I by a single "Reconcile" module (but uploading is a separate process). The first stage, that of "Zone Validation", is new. With automated labeling available, that step was seen as the subsequent inspection and correction of both zone boundaries and labels. Its inclusion is necessitated by the belief that the automated segmentation and labeling systems will not be 100% reliable, due to imperfect OCR data, or the use of AI systems with initially sub-optimal performance. Potential sources of AI difficulty include undersized training sets, incomplete coverage across the large, stylistically-heterogeneous collection of journals, or

Table 1. Early MARS II Workflow Design [1]. Manually-operated modules (non-daemons) are in **bold**. While details of the number, names, and order of modules have since evolved, and a number of processing aspects refined, the overall concept remains generally valid.

Station	Purpose
Scan	Enter a new journal into the MARS II system, then scan the first page of each relevant article within.
Segmentation	Locate paragraph zones within each scanned image.
OCR	On each image, perform per-zone optical character recognition (OCR). As with MARS I, the Prime Recognition 5-engine OCR system is employed, but writing to the database, not to ".pro" files.
Spell Check	Remove doubt about low-confidence OCR characters if they appear within words found in a special dictionary
Zone Validation	Associate a type, such as "Title", with each zone of interest.
OCR Validation	Typed correction of OCR errors. This could be per image and zone, as in MARS I, or per character across images using "carpet" correction.
Record Validation	Handle all remaining data entry or correction tasks. Then upload to the pre-MEDLINE machine (for further specialist indexing, then entry into MEDLINE)
Archive	Periodic off-loading of production database

inadequate zone-feature recognition and categorization. In any event, errors in auto-labeling, if not caught early, propagate downstream to cause more corrective work for the final record validation or reconciliation operator.

Comparing Zone Validation with the tool design in the previous section, it was quickly seen that this proposed module shared a preponderance of goals and features, particularly since one type of "tabular form" was database tables. However, since the form of the database was still in development, saving data in tabular form to the file system was also needed.

The main conceptual change required by the production role of Zone Checker was to Step 2, which now must also allow the simpler alternative of reading in zone location (generated by Segmentation) and OCR data, instead of performing its own calculation.

2.4 Two Roles, One Tool

Combining the two roles seemed both parsimonious of development effort and functionally synergistic. Thus, from the outset, Zone Checker was shaped towards becoming a possible MARS II component. This duality of roles had drawbacks and advantages. One advantage was that CEB researchers, as users of this tool and more accessible than the production personnel, provided early and on-going informal usability testing, and a stream of suggestions for enhancements.

2.5 To a New Step 2

The research role's Step 2 called for incorporating border-finding and segmentation; the thought was to upgrade and integrate some existing libraries. When exploratory efforts during the first implementation period revealed severe technical and legal hurdles, this goal was dropped, substituting:

2a. Read in zone locations and corresponding OCR data (characters, attributes, and bounding boxes) from a source external to Zone Checker.

The initial source for all this information was ".pro" files generated by the Prime Recognition OCR Server, with "auto-zoning" enabled, so as to include the segmentation task.

From the point of view of the production role, this information would instead be found in the database, generated by the OCR software daemon. CEB's current version of the latter, "Prod", also uses the PR OCR Server with auto-zoning.

As mentioned earlier, the early database design did not indicate where auto-labeling would occur. For convergence, it was decided to add another optional step to Zone Checker:

2b. Algorithmically guess each zone's label type.

For MARS II production, doing the (2b) work within Zone Checker was seen as a provisional convenience, predicated on the assumption that it would take no more than a second or two per image, and thus wouldn't slow down an operator significantly. A quite recent direction is to relocate the auto-labeling to a new unattended process earlier in the workflow, which also would incorporate a new zone-boundary correction phase.

Figure 1. The MFC Demo Sample Program, MFCdem32. Shown are two open general-purpose images, and a few of the available image processing operations.

3 Early Design Choices and First Implementation

3.1 Selecting an Imaging Library and Starting Application Framework

Beginning with the foregoing concept of Zone Checker (basically a TIFF viewer and label editor with both file and database storage), a review of commercial Win32 imaging libraries was undertaken, looking particularly for components that support annotations atop scrollable images[5]. Lead Technologies' "Lead Tools Pro" [2] in dynamic-link-library form was selected, which advantageously had C++ wrappers (albeit thin ones) around its C API. In addition, a significant amount of sample source code was provided. In particular, "MFC Demo" [Figure 1], a standard DevStudio-wizard-generated code skeleton that had been fleshed out to encompass most of the Lead Tools API, became the starting point for Zone Checker. It was attractive because it immediately provided an extensive core of image file processing functionality. However, it lacked annotation features, which were hand-merged from another sample program [Figure 2].

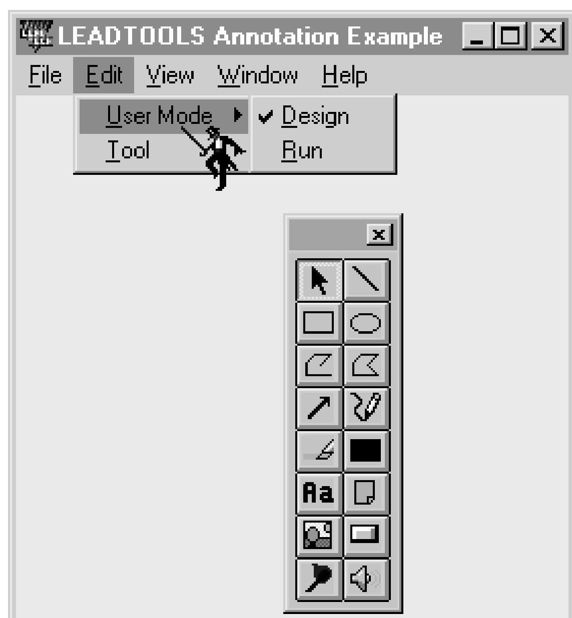

Figure 2. The Annotation Sample Program, Annot32. The floating toolbar shown for drawing annotations is discussed in a later section. Most of these annotation types are specializations of rectangles, such as hotspots, buttons, text boxes, highlights, and redactions (black-outs). The "run" and "design" modes became the internal basis for Zone Checker's "Adjust Labels" and "Select Zone" modes, respectively.

---

[5] In-house TIFF reader/writer code from another project was also considered as a starting point, but passed up because its conversion from 16- to 32-bit Windows was not complete in early 1997, nor was there annotation support. Note that Zone Checker development predated availability of Wang Imaging for NT, used in OCX form by some other MARS II modules.

3.2 Going with the Flow

With a compendium of third-party code such as the Lead Tools library, there are always some things that are made trivial or straightforward for the programmer, and other things that are difficult. Part of initial exploration is to identify these fairways and mine fields, and try to drive the design in such as way that the needs of the application can be satisfied by operations within the friendly landscape. This is as in contrast to a design philosophy, perhaps more appropriate for mass-market commercial software, in which the aesthetics and affordances[6] are prespecified in the absence of implementation considerations.

3.3 The Trade-offs of Featurism

Zone Checker's dual roles entailed supporting both an emerging database and existing file-based storage. As a "Swiss Army knife", this flexibility allowed it to be used as a go-between, for instance, to load historical file-based data into the database. The cost is that of software bloat [3], not just in terms of memory and hard drive footprint, but in terms of presenting users with a rich but complicated feature set, some of which are unused by certain user classes. An on-going effort involves eliding bloat wherever possible through feature re-packaging, hiding, disabling, or deletion.

For instance, the "Annotations" toolbar is very helpful for the developer in exploratory try outs of different on-screen representations. But, because marks drawn with it have no tie-in to OCR data, it is not helpful for most end-users. A new menu item was therefore introduced to toggle its presence[7]. This toggle is inaccessible to production users, for whom the toolbar is always hidden.

3.3.1 How Many Page Views?

"MFC Demo" provided a multiple-document user interface (MDI), so the first question was whether to reimplement it as a single document interface (SDI). It was left as MDI, mainly to minimize up-front development time and risk. Also, it was known that occasionally both the first and second pages of a journal article were scanned, because the abstract ran over to the second page. In such cases, the ability to see and touch both pages at the same time could be helpful. However, sticking with MDI caused later development time penalties, in working with the more complex doc-view internal structure and in making the seldom-needed multiple-document aspects less intrusive to the user. Most other MARS II modules that display images use form-based SDI.

[6] That is, GUI ways of offering control of functionality.

[7] This toolbar refuses to be hidden in a Windows API sense, so it's moved past the screen edge to hide it.

3.3.2 The Triage of Imaging Features

The broad outline of MFC Demo's user interface was that provided by the DevStudio/MFC wizard, and initially retained by Zone Checker (until the later make-over of Section 5.1). But a few wizard-provided features, such as the most-recently-used file list, were removed because they seemed hard to extend to database interactions.

While MFC Demo ties "File/New" to a TWAIN-compliant local scanner, this capability was dropped from Zone Checker as unneeded and a potential support headache. Instead, the early Zone Checker user invoked a standard File Open dialog to choose an existing MARS I image, read-in via Lead Tools TIFF decompression. To enhance bitmap readability, the default display mode of scalable images was changed to scale-to-gray; the bitonal file image itself remained unchanged in normal usage.

Of the many image processing algorithms found in MFC Demo, those clearly irrelevant to Zone Checker (e.g., artistic effects and "slide show" multi-image transitions) were dropped. A few that seemed particularly germane (e.g., deskew, flip) were made more prominent in the top menu structure. Conversely, ones anticipated to be of infrequent use were buried deeper into the menu hierarchy under a general title of "Specials", an example of a bloat-hiding strategem.

MFC Demo had zoom menu items, but they weren't well matched to our needs. An early change (that matured through a number of improvements) was to introduce a separate "Zoom" bar.

3.3.3 Juggling Multiple Purposes with Setup Property Pages

Since Zone Checker had both research and production roles, it was obvious that configuration control was important. This first took the form of File/Open Setup and File/Close Setup menu choices. As its name suggests, Open Setup controlled what additional processing steps occur associated with opening each image. Initially, Open Setup provided a single dialog for locating the corresponding OCR file (e.g., 1.pro for 1.tif), the only source of zone-specific OCR information at the time. This file might be in the same directory as the image, or in a directory location given in the early database. The contents of the property page was soon expanded to hold database log-on and database vs. file-system-only choices. Then a second tabbed property page appeared for the feature discussed next. Close Setup saw similar incremental growth.

3.4 Communicating Transient Operations During Image Opening

As each image was opened, a configurable series of processing steps, of perhaps several seconds total duration, need to be applied to it. What should be displayed while this occurred? While a modal hourglass cursor, possibly-modeless progress bar, or other animation were considered, a more informative display was sought. A common alternative, text in the status bar, would mutate too quickly to be discernable. Instead, a custom dialog briefly appears [Figure 3].

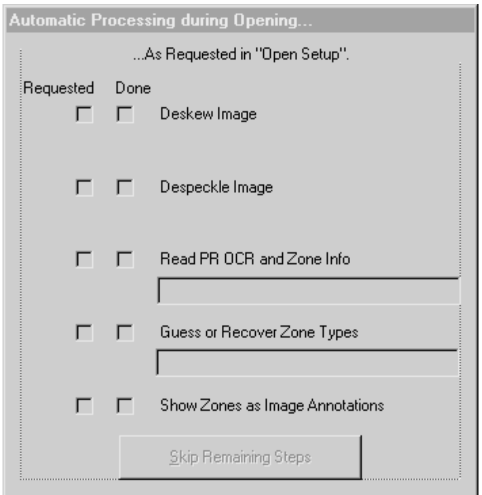

Figure 3. Steps during Image Opening. The left column shows requested steps, the right those that are completed, with processing in top to bottom order. This shows the current version of this dialog; the original didn't have the extra text fields, which indicate, for instance, whether labeling is done by the built-in rules or by reading an external file. There is a setup property page of similar appearance, with a single column of checkboxes, in which steps may be requested or not. The despeckle and deskew steps are seldom needed; MARS II images will be already deskewed. It is the next-to-last labeling step that is most frequently toggled in research activities.

3.5 Enumerating Zone Types – Rich versus Minimal

MARS I reports a dozen-odd document field types to MEDLINE. On the other hand, one alternative method of document input to MEDLINE, electronic submission of structured documents directly from publishers to NLM, defines about fifty SGML tags. As part of developing Zone Checker, a rich zone descriptive system was defined, closely modeled on the latter but with some additional distinctions. This is flexible and allows AI training to categorize types that may be more important for exclusion than inclusion. Nevertheless, a minimal set is usually of most immediate interest.

3.6 How should a Zone Appear? The First Zonescape

Early on, the Lead Tools "rectangle annotation" was chosen as the representation for a zone. This is aligned with the image edges (although programmatic rotation is possible). Since the zone boundaries given by OCR were also so aligned, this was convenient from a structural standpoint. From the GUI point of view, the rectangle is much easier to manipulate than the main alternative, a polygon annotation. The latter is more flexible, allowing, for instance, picking out a particular few sentences from a paragraph, but positioning all the control points would be burdensome for the user, and tricky for the developer in relating to underlying OCR data. Instead, there could be multiple rectangles of the same type, linked implicitly by relative order or explicitly. Furthermore, a zone denoting a small snippet of text might be overlaid on a larger zone.

It was decided that each zone could be only of one type, and zone splitting or overlays employed as needed to work around any restrictions this might impose. Each zone type is either "major", represented by a particular solid color ("translucent" in Lead Tools parlance), or "minor", with "clear" interior and a colored border, most suitable for overlaying on major zones. An example is shown with major zones [Figure 4]. A fixed palette of colors relating to particular zone types (later called a "zonescape") was developed. Colors selected were spread out in a spectrum, with zone types varying from red to purple corresponding to where they most commonly appeared on a page. Most colors were fairly bright. For differentiation, colors for unknown zone type or body text were pastel.

Zones that were most likely garbage were specially rendered to be unobstrusive. For instance, type "White-out Blem (nontext)" means that the zoned portion of the original page has no actual text, and typically no actual graphical content either. If the OCR did report any text, it is a misrecognition; for instance, it is not uncommon for zones around border or gutter shadows to have phantom characters like "i" or "I" in them. Zone Checker visualizes this distinction with a subtle hint: With no OCR text, the zone is shown in light gray crosshatches on opaque white; otherwise, the crosshatches are golden.

3.7 Implementing Zones

A great deal of early effort was being able to read in the ".pro" data, associate it with internal data objects, and represent it on screen using the rectangle annotation.

Atop a visible "gell" zone is a totally transparent "hot spot" zone of the same size. A click on the hot spot routes an event (including a hot spot ID) to a Lead Tools-specific callback function within application code. In our case, the hot spot ID allows discovery of the corresponding "Zone Object", the overall manager of a specific zone's annotation, OCR data, and other information. This information is used to position the zone-type selection popup menu, and is revised if the user changes the zone's type, which causes an immediate change to the zone's color.

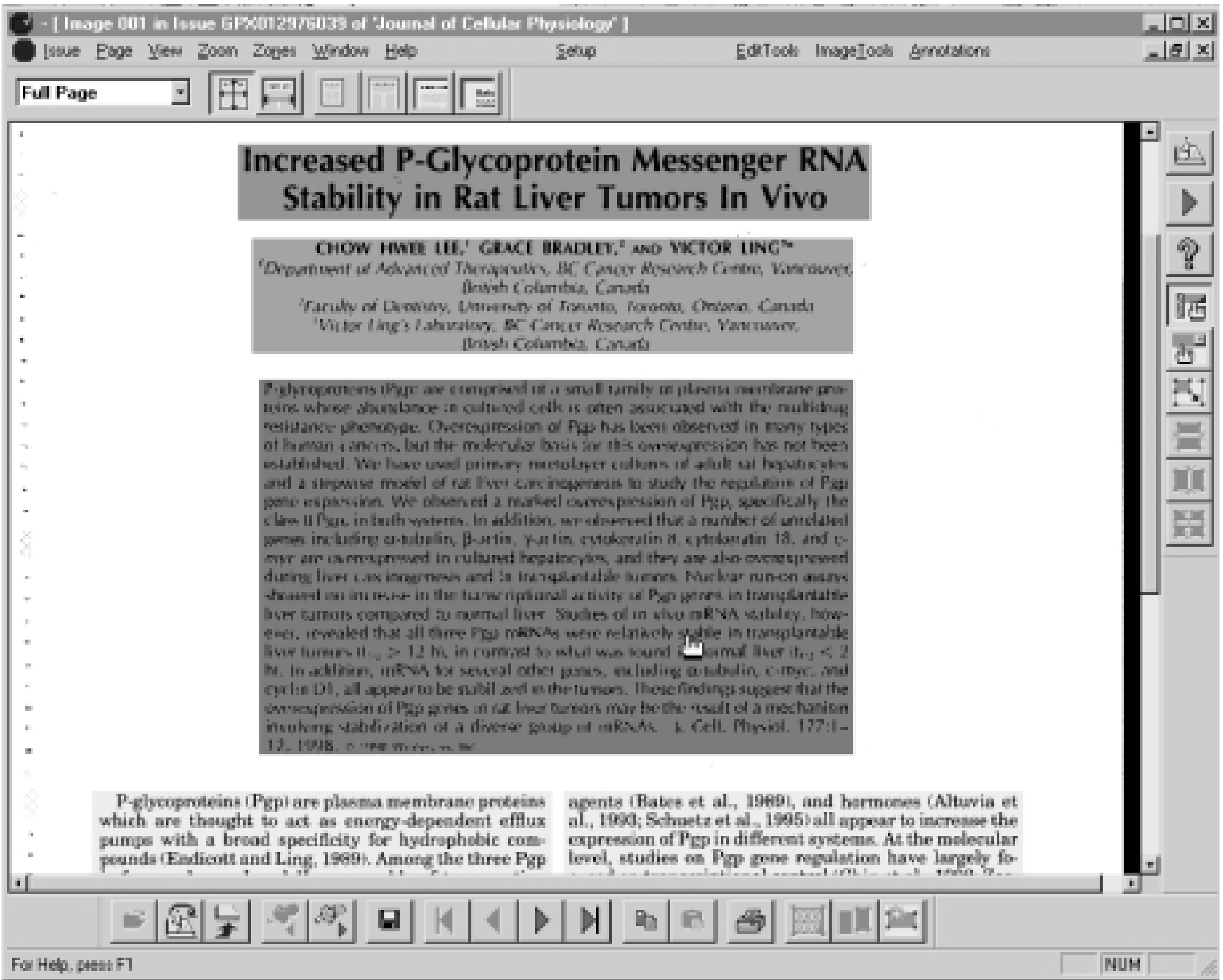

Figure 4. The Current Appearance of Zone Checker. Zone coloring is discussed in the main text. Note in particular the small "White-out Blem" zones near the left edge, with their faint crosshatches. The frame shows three toolbars, "Zoom" at top, "Main" at right, and "Specials" at bottom. The latter is usually hidden. The buttons in Main are, from top, Open Issue, Next Page, Help, the three mutually-exclusive modes ("Adjust Label", "Set Reading Order", and "Select Zones"), and additional buttons that become enabled in select-zones mode: Split Horizontally, Split Vertically, and (not yet implemented) Merge.

4 Under the Hood

As with many software projects, some time-consuming aspects of Zone Checker development had a relatively modest impact on the user interface.

4.1 Automatic Guessing of Zone Labels.

Zone Checker served as a research testbed for "first generation" rule-based algorithms for automated labeling of zones [Appendix]. If enabled, these built-in C++ rules would activate as each image was opened and the corresponding OCR data automatically read in.

4.2 Saving Zone Features and Labels

It became valuable to be able to save the zone labels, as well as certain of the calculated features, to files, respectively called ".lab" and ".zon" files. These could then be used to train external AI labeling systems, such as a neural net [4]. Later, it became possible for Zone Checker to read .lab files as well; and analogous read/write actions using the database as the persistent label store were added. The writing of this information occurred as each image was closed, which necessitated adding more property pages to Close Setup.

4.3 Interfacing to the Prototype Database

To get images and store results, Zone Checker connects automatically on start-up to a MARS II-specific database, unless configured to use only the file system. The initial test database for Zone Checker was a local MS Access one, accessed via MFC's ODBC class wrappers. But within six months, Zone Checker had migrated to a small SQL Server 6.5 database elsewhere on the LAN, populated with MARS I images and information, suitable for testing and initial profile development. ODBC was phased out in favor of RogueWave's DBTools.h++ [5] with its associated SQL Server driver. Subsequently, the design and content of the database continued to evolve as many complicated design issues were resolved. Rogue Wave/C++ wrapper classes for most database tables were coded and incorporated into a shared library used by Zone Checker and other modules. Windows NT became the platform target for both database and clients.

5 Recent GUI Improvements

A number of usability enhancements have occurred within the last six months.

5.1 Redesign of Main Menus and Workflow

A GUI redo [Figure 5] streamlined and refocused the application towards production. The items within the main “File” menu were parceled out into separate new “Issue”, “Page”, and “Setup” topics. The latter merged File's setup options [Section 3.3.3.] with those (seldom appropriate to change, such as scale-to-gray) of “Preferences”, and sequestered them from production operators. To reflect their non-production status, “Edit” and “Image” were renamed “EditTools” and “ImageTools” and moved to the right side of the window frame; EditTools got the various print-related functions from “File”. The new “Zones” menu (used as an example in Figure 6) absorbed general-use operations from “Annotations”, leaving behind researcher-only functions. The main toolbar was split into “Main” and “Specials” (as shown in the earlier figure), and a show/hide feature for the latter added to “View”. Subsequently, View became the repository for general-user options, such as mini-icons [6] on the menus [Figure 6] and toolbar button size.

An important workflow improvement was to move away from generic {File/Open, File/Close} processing to an {Open Issue, Go to Next Page} paradigm for both file-system-only and database configurations. But the original random-image-access method is still available for special research purposes, now as Page/Open.

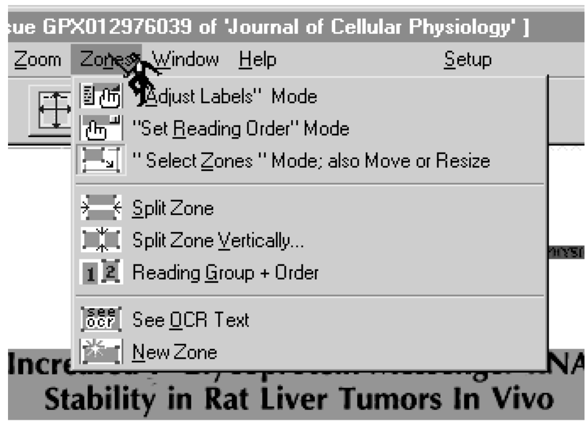

Figure 6. Mini-Icons on Menus. Every menu item that has a corresponding toolbar button is decorated with a small version of the same image. The user exercises View/Options to turn this feature on or off.

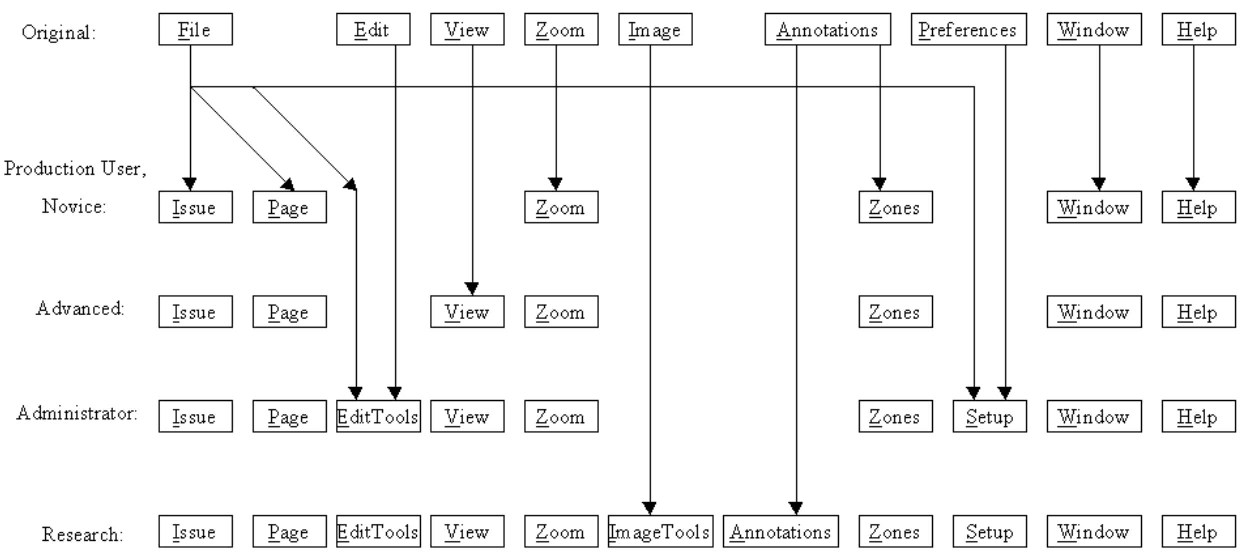

Figure 5. Redesigning the Top Menu Structure and its Accessibility by Class of User. The top row has the original left-to-right menu order. The “novice” versus “advanced” production distinction is so far only theoretical. A domain group was established for research users, and the current user’s membership within it checked at Zone Checker startup. Appropriate features are disabled for non-researchers. In addition, the Setup menu solicits a special administrative password. This approach was taken so that a production user with a setup problem could ask the local manager to intervene without requiring the user to log off. Note that operations with the MARS II database require user pre-registration.

5.2 Ways of Visualizing Zone Types

For a long time, the single fixed zonescape (label palette) discussed earlier was the only feedback to the user as to zone type. The total palette mapping could be viewed within on-line help or as a color print.

5.2.1 A User Designed Zonescape

Users may differ in their color acuity and preferences. To accommodate this, as well as facilitate usability testing to optimize the default zonescape's colors, a second, user-definable zonescape was invented. Management of these two zonescapes occurs in the new View/Options dialog [Figure 7]. Further zonescapes may be added in the future.

5.2.2 Zone Types Shown as Text

When in "adjust label" mode, it was felt that textual feedback in addition to colors would be helpful. Figure 8 shows one very early idea. Another was placing a check next to the zone-side popup menu item (but this is less useful with a multilevel menu). A third possibility was having a toolbar palette of color swatches and zone labels, analogous to our zonescape-setting property page, but always visible.

The direction pursued was instead to have "fly-over" help when moving the cursor over a zone. While the text, e.g., "Abstract", could have been put into the status bar, it was decided that a tool tip rectangle immediately below the cursor would lessen eye transversals. Since this appears and disappears automatically, and is repositionable by moving the cursor, it doesn't get in the way of the underlying bitmap text.

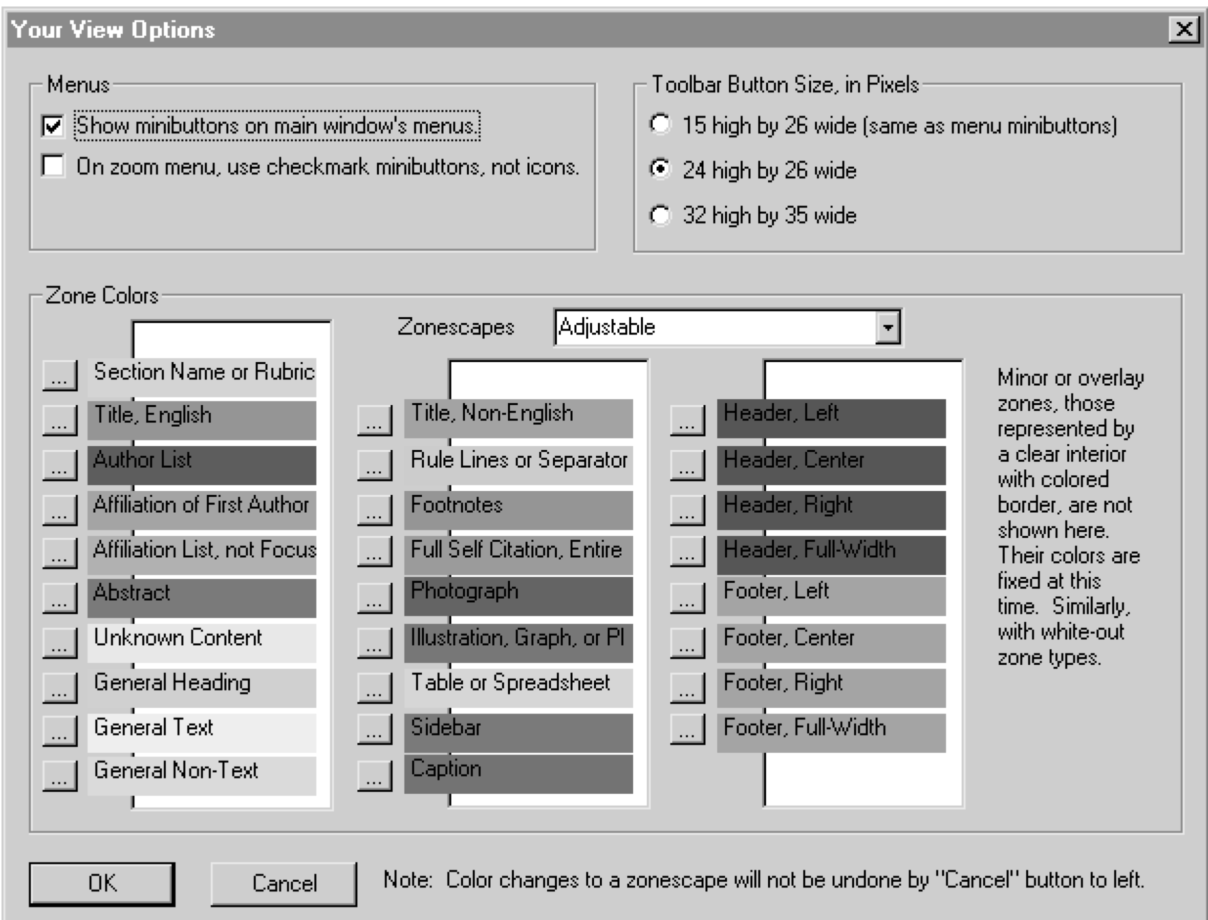

Figure 7. Zonescape Management. Only the set of 27 "major" zones (those with solid colors) are shown as swatches that can be manipulated at this time. Pressing a button to the left of a zone brings up the standard Windows "color chooser". A new color can be selected therein via several ways, unless the zonescape is the read-only default one. The color chooser has a set of "custom color" boxes, which here is used as a most-recently-visited list. This makes it easy to copy colors between zone types or between zonescapes.

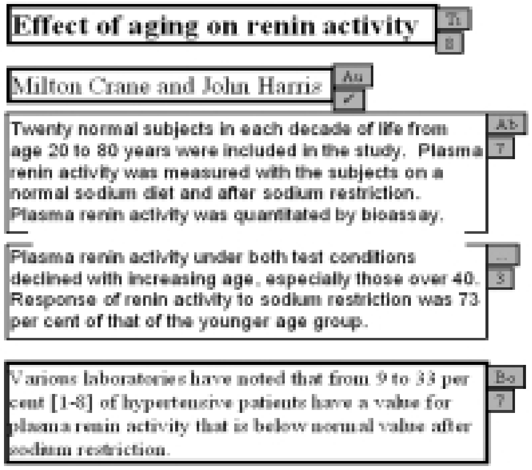

Figure 8. The Early "Ears" Concept. In this mockup, each zone was decorated with a two-button "ear". One button holds a zone type abbreviation: "Ti" for title, "Au" for author, "Af" for affiliation, "Bo" for body text; etc. A suffix digit could also appear, for instance, "Au1", "Au2" if there were two author zones separated by a non-author one. "…" was used to indicate, "this zone is a continuation of the previous zone's label." Continuations were further indicated by leaving the abutting top and bottom borders open, except at the corners. (This style is not part of the available Lead Tools repertoire; perhaps it could be drawn by overlaying an opaque white line on a rectangle's border.) The other ear button showed a labeling confidence level, in the range 0..9. (Labeling confidence values are not yet available.) An ear is also a hotspot, so that the operator can override the guessed value. In such a case, the confidence digit is replaced by a checkmark. Current Zone Checker handles labels and reading order as more distinct modalities than the ears concept. But the implemented reading order display, discussed next, is stylistically similar.

5.3 Reading Order

It's often necessary to determine the "reading order" among certain zones, particularly those of the same label type whose OCR text is to be combined downstream. This is particular true for same-type zones that are not adjacent, such as those containing text flowing from one column to the next, and thus not candidates for merging. The normal OCR system does not deduce full-page reading order. There is an optional mode to use Xerox's Textbridge engine to attempt such a sort, but this is reportedly problematic. In general, even with content understanding, it is difficult for humans to always agree on the proper reading order encompassing <u>all</u> text elements on a page.

Zone Checker would instead support manual partial ordering, by letting the user define one or more zone groups, and set the order within each. Zone Checker would automatically provide a default full-page group. Within this group, ordering is top-to-bottom by top zone edge, then left-to-right order by left edge. This ordering calculation is performed when zones are first read in, and on subsequent pertinent events such as zone splits. Placing a zone in a user-specified group in effect hides the default group ordering.

5.3.1 Initial Interaction Approach

The first implementation was as a new function within the existing "select zones" mode. The user selects a zone, then hits a "reading order" button on the main toolbar. This brought forth a dialog box, with two fields to view and set a "group number" (greater than "1" for user-defined) and order within group. As this proved extremely tedious, a third mode, "reading order", was conceived. Here, every zone would have its own widget. This would display on its face the value of its current reading order, and could be clicked upon to alter it. Displaying reading order as text seemed more straightforward to present to users than alternatives such as drawing arrows between zones.

5.3.2 Widget Choices – On the Button

As widget candidates, Lead Tools provides a number of annotation types, such as stamp, note, or the familiar rectangle. But most compelling seemed the button annotation with the usual Windows beveled-button look. The opaqueness of the button is helpful. Another advantage is that it intrinsically responds to click events, unlike rectangles for which one must maintain a hotspot separately. (Internally, button events would be associated with a particular zone rectangle by using a programmer-defined numeric "tag", just like the existing zone hot-spot/gell implementation.). A button's text and the text color can be changed, but the font and its size are that of the current system font. The background color is the current Windows frame color.

Like all annotations, the button object is inserted into the "annotation container" that holds the zone rectangles. Thus, zooming the image, which re-scales all container objects, resizes the button itself. But, as noted, not the text. Thus, the button size must be chosen to be large enough so that centered text won't be clipped at the edges when an image is zoomed out and the button appears tiny. The size specified is currently independent of text. (Further improvement to button sizing, possibly with font metrics lookup, is possible.)

The default zone-relative placement of buttons should obscure as little "important" bitmap text as possible. It might be inside or immediately outside the border. The worst place inside is probably the upper-left corner. The right edge appeals over the left, since text is usually left-justified, but often not right-justified. Arbitrarily, the buttons are placed in the upper right corner; lower right or the centroid might work, too.

The use of numerals for both reading group and order within group was problematic. After experimenting with a "[2] 3" notation, capital letters were substituted, leading to a "A3" notation for group A, order number 3. "A" is the first user-selected reading group (internal number 2). A missing letter denoted the default group, e.g. "3". The dialog box was altered to reflect this style (Figure 9). On the buttons themselves, the default group text is in black, the user-set groups in red. (It might be interesting to color code each user-set group differently, to see if that is helpful or confusing.)

5.3.3 Expedited Interaction

Within the "reading order" mode, it would be better to just touch each zone in turn to set the order, and let the system assign the order numbers. Rather than add this to the button functionality, it is touching the non-button part of the zone that does this. Then button presses are merely for corrections, a slower but infrequent process.

This is a little complicated, given our two-part reading-group/reading-order bifurcation. It was decided to program the interaction this way: the first time one enters reading-order mode for a given image, zone touches cause sequential numbering within group "A". One leaves that reading group by going to one of the other modes besides reading-order. Re-entering reading-order mode moves to group "B". This approach, while a little awkward, avoids introducing more controls, and seems reasonable when in the projected usage, few images will have more than one user-defined reading group.

If a zone is tiny, the fixed-size button placed strictly in the upper right corner totally obscures it, denying access to any non-button zone area to click on for order setting. A related problem: a small zone at the bottom or left edge has its button clipped at the image bounds. For these cases, button placement is shifted sideways, and/or aligned with a different zone corner. Nevertheless, each button is positioned independently, so buttons of adjacent zones can overlap. This may not be a problem in practice, since tiny or near-border zones seldom require user concern about reading order.

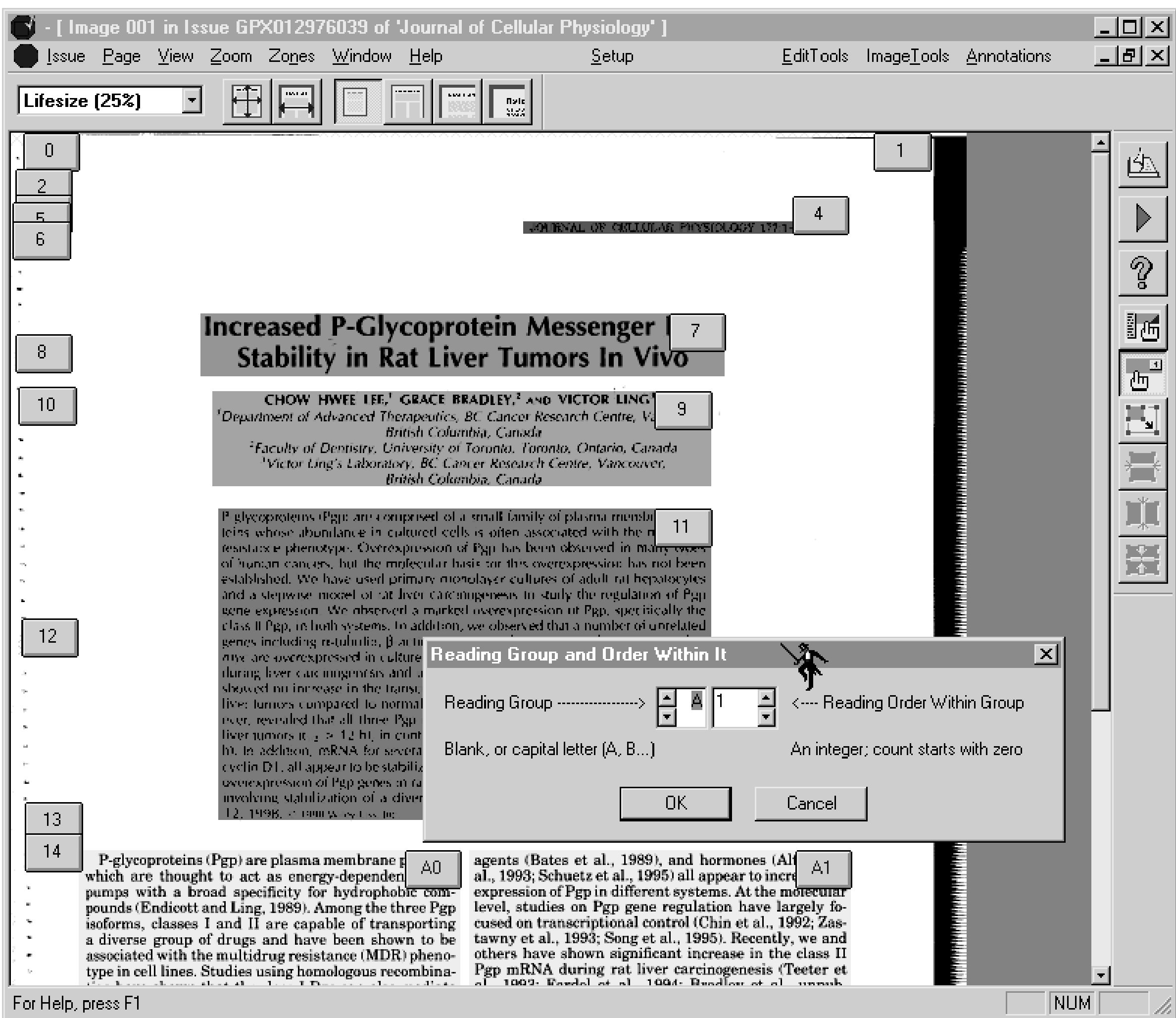

Figure 9. Displaying and Setting Reading Order. In this example, the two paragraphs near page bottom were assigned their own reading group "A" using the "expedited interaction" method of clicking the body of each in turn. If the user then had second thoughts about the right-most paragraph, clicking on its button brings up the dialog box shown, for correction. Selecting "blank" in the reading group field makes the order field read-only and automatically fills in the default ordering value (evidently 15 or 16 in this case).

6 Further GUI Developments in Progress

6.1 Correction of Zone Types

In adjust-labels mode, clicking on a particular zone brings up, immediately to its left, a popup menu. This predominately 2-level menu offers the entire set of possible zone types. Again, we face the issue of a rich versus sparse representation. With a large set of available zones, finding the ones in this menu of most importance because more problematic. In the long run, making this menu's contents dynamic, thus settable at runtime, instead of compile time, would allow both flexibility and concision. A tree control has been prototyped that reproduces the rich menu hierarchy and would allow a user to mark each zone type therein as skipped (hidden), optional, or required.

6.2 Zone Splits and Other Manipulations

From a GUI-design perspective, the area of greatest difficulty may be in splitting zones horizontally or vertically, or otherwise extracting portions of a zone. Splitting cannot happen at any arbitrary pixel, but only where the underlying OCR data permits. Furthermore, for vertical splits, it appears that some awareness of the underlying OCR data needs to be brought to the user's attention in a dynamic way. Zone "direct manipulation" [7] for splitting may be better done via mediating devices, such as sliders, that can be easily paired with OCR data display. Mediated horizontal and vertical splitters are being prototyped.

In Zone Checker, it is possible to create a new zone from thin air, and to alter the shape (or delete) existing ones. However, the degree of integration between PRO-specified zones and resized/new zones is rather weak. Specifically, changing a zone's shape has no effect on the extent of its OCR data, nor do new zones, used as overlays, inherit OCR data from their parent zone. This is an area for future consideration. Of more immediately need is the ability to merge adjacent zones. This will be non-trivial when they are side-by-side and individual OCR text lines must be aligned and appended.

7 Conclusions

Zone annotations are convenient "handles" for manipulating the underlying OCR data, and several ways of doing so have been presented. In this matter, as in many software design issues, much of the effort goes into finding the right balance between contending goals. We have retraced one exploration, that sought the balance between production and research needs, between a focused versus extensive feature set, and between engaging design concepts and implementation pragmatics. The tool thus created has multiple prongs and enjoys multiple purposes in furthering the next version of MARS.

References

[1] Ford, Glenn, *MARS Technical Specification 0.0B*, CEB internal document, 12/2/1996

[2] Lead Technologies, *LeadTools API Manual for Pro Express*, v. 7.0, Charlotte, NC, 1997.

[3] Kaufman, Leah, and Brad Weed, "Too Much of a Good Thing? Identifying and Resolving Bloat in the User Interface", *SIGCHI_Bulletin*, **32** (4), ACM, Oct.,1998, pp. 46-47.

[4] Le, Daniel, Jongwoo Kim, Glenn Pearson, George Thoma, "Automated Labeling of Zones from Scanned Documents", *SDIUT'99*, April, 1999.

[5] Rogue Wave Software, *DB Tools.h++ User's Guide and Tutorial*, Corvallis, OR, 1998.

[6] DiLascia, Paul, "New Interface Look: Cool Menu Buttons", *Microsoft Systems Journal*, Jan., 1998.

[7] Shneiderman, Ben, *Designing the User Interface*, Addison-Wesley, 1987.

[8] Hauser, S., personal communications, CEB, 1998.

[9] NLM, *List of Serials Indexed for Online Users*, ISSN 0736-7139, 1997.

[10] Kanungo, T., G. Marton, O. Bulbul, *Paired Model Eval. of OCR Algor.*, LAMP-TR-030, Inst. Adv. Comp. Stud., U.Maryland, College Park, Dec. 1998

Appendix. Built-in Auto-Labeling

Experience with these mostly-journal-independent "first generation" rules (Table 2) indicate that they were quite good at detecting white-out zones, albeit with some mislabeling of in-border page numbers as such. They were moderately good at the zones of most MEDLINE importance, depending on the degree to which the journal style is a common one. For instance, in a brief test with 10 autozoned page images from a single "easy format" journal issue, all title zones were correctly auto-labeled, as were 9 of 10 author zones. However, 3 of 10 affiliations were tagged as authors. The main abstract zones were correctly tagged in 10 of 11 cases, but 3 abstracts had a short last sentence, separately zoned, that was missed. The two mentioned problems might be ameliorated by more extensive interzone comparisons, and by further enlargement and refinement of the word lists beyond those in Table 2. The separately-developed "second generation" system [4], which, for example, has a word list for Affiliations based on historic data that is an order of magnitude larger than Zone Checker's, shows the effectiveness of these strategies. Other aspects that were nascent here and full-bodied in [4] include a numeric confidence value associated with label discovery and a multi-phase convergence upon the set of labels for a page. But perhaps most trenchantly, the advantages of aligning algorithms more closely with specific journal styles is manifest.

Table 2 (a-d). Built-in Zone Feature Recognition and Labeling Rules.

When an image's OCR and zone-location data is read into Zone Checker, the default label given to all zones is "Unknown Content". An quick software screening sees if this consists only of MARS I's manually-zoned title and abstract. Otherwise, auto-labeling analysis begins. Each zone is handled independently until the very end. For each, a group of quickly-calculated features are found in unnormalized and normalized forms. (The rightmost column of Table 1 in [4] enumerates those normalized features selected for export to a neural net system; others calculated include zone order, number of words, and number of initials, e.g., " A."). Next, a series of if-then-else tests proceeds (as given by row order in the subtables below) until a label is assigned. The most important labels for MEDLINE are shown **in bold**. A final limited revision step uses zone interdependencies: if no abstract was found above, one of the unknown or general-text zones in a particular part of the page may be relabeled as abstract.

a. Zones with Few or No Valid Characters. A long, thin zone without characters may be a rule line or similar separator. Otherwise, it's probably a "white-out blem", typically a black gutter or page edge artifact. "White-out Text" is most often text fragments on the facing page across the gutter. A page number is another possibility.

Constraints on OCR Text in Zone	**Constraints on Zone Location**	**Assigned Zone Label Type**
No characters	Overlaps central 2/3rds of image; > 1" long and < 1" wide (either orientation)	Rule Line/Separator
	otherwise	White-out Blem (non-text)
Entirely low-confidence characters (< 7 out of 9)	Fully within 1" of any image edge	White-out Blem (non-text)
Single numeral	Ditto	Page Number
Other single character	Ditto	White-out Blem (non-text)
> 1 characters	Fully within 1" of the left or right edge	White-out Text
	Fully within 1" of the top edge or 2" of bottom	Page Number

b. General Cue Word Matches. The matching here and in (c.) is intentionally case insensitive. A "delimited" cue word or phrase is on a line by itself, or followed by a space, semicolon, period, colon, or dash. A few dozen common biomedical headings are recognized, e.g., "chemicals", "enzyme assays", "experimental techniques", "growth conditions", "materials", "media", "methods", "mice", "production of", "reagents", "strains".

OCR Text in Zone	**Constraints on Text**	**Assigned Zone Label Type…**	
		…if one line of text	**…if multiple lines**
"received" or "first received"	Begins zone; Can skip any one prefix character, like "("	Date Received Or Accepted	
"revised" or "accepted"	Contained in zone		
"abstract"	Begins zone; Delimited	Abstract Heading	**Abstract**
"keywords" or "key words"	Begins zone; Delimited	Keyword List	
"introduction"	Begins zone; Delimited	Introduction Heading	General Text
Certain biomedical headings	Begins zone; Delimited	General Heading	General Text

c. Assignments for Central Zones. These final assignments are for zones that width-wise overlap the middle 2/3 of the image. If a label is still not assigned at the end of this process, it is left as unknown.

OCR Text in Zone	**Constraints on Text**	**Additional Constraints on Zone Location**	**Assigned Zone Label Type**
"case report", "case reports", "comentary", "editorial", "opinion", or "notes"	Begins zone; Delimited	In top 25% (2.75") of image	Section Name (if single line of text), or Title
--	Average point size of top-confidence characters > 14	In top 40% (4.4") of image	**Title**

c. continued next page

<table>
<tr><th>OCR Text in Zone</th><th colspan="2">Constraints on Text</th><th>Additional Constraints on Zone Location</th><th>Assigned Zone Label Type</th></tr>
<tr><td>Year, month (or its abbrev.), “journal”, “j.”, “volume”, “vol.”, “number”, or “no.”</td><td colspan="2">Contained in first line; Delimited</td><td>In top 1.5 inches*</td><td>Header*</td></tr>
<tr><td>See Figure Xd</td><td colspan="2">See Figure Xd</td><td>Top edge below 1"; bottom above 50% (5.5")</td><td>Affiliation</td></tr>
<tr><td rowspan="2">“to whom correspondence” or “author to whom correspondence” or “corresponding author”</td><td rowspan="2">Begins zone; up to 3 prefix characters can be skipped; Delimited</td><td>> 3 lines</td><td>Top edge below 6"</td><td>Correspondence To</td></tr>
<tr><td></td><td></td><td>Affiliation</td></tr>
<tr><td>Initials (e.g., “ X.”) or commas (or “and” counted as if a comma)</td><td colspan="2">On average > 1 initials per 20 characters or > 1 commas per 20. Not more than 1 occurrence of “and”</td><td>Fully in top 40%</td><td>Authors</td></tr>
<tr><td>Highest confidence characters</td><td colspan="2">Average font size > 8 and < 13</td><td></td><td>General Text</td></tr>
</table>

* further subcategorized by size and position as Left, Right, Center, or Full Header

d. Cue Words for Author Affiliation. Affiliation zones are recognized by a coarse location screening followed by cue word matching, with the heuristic that there must be at least two cue words found on average per OCR line. A match constraint is that the first word of the OCR text must be capitalized (except suffix matches). String routines were built that match in the face of OCR errors due to common character misrecognitions [8]. Using reference books, a specialized cue word list was created, made up of the following groups (with varying degrees of coverage of non-English words), each with at least the number of items shown in the right column. NLM-indexed biomedical journal titles were used as a source for biomedical fields of study [9].

Category	**Representative Subcategories, and Examples in Quotes**	**Items**
General nouns for geopolitical entities	“Commonwealth”, “State”, “Republic”, “Ville”, “Town”	40
Specific geopolitical descriptors	Country names, with variant spellings	300
	States or provinces of US (including 2-letter abbreviations), Canada, Mexico, and China.	300
	Major cities (chiefly world capitals)	
	Common town suffixes (mainly US). …“ton”, “mouth”, “stad”	50
General nouns for geographic features. These often appear as part of a multi-word town or organizational name	Bodies of water and shorelines, e.g., “Rive”, “Porto”, “Springs”	180
	Terrain (“Mt.”, “Valle”, “Serra”)	100
	Human constructions (“Depot”, “Church”, “Mill”)	40
General geographic adjectives	Compass directions (“Western”, “Ost”, “Southeastern”)	40
	Relative position or size (“Upper”, “Outer”, “Mid”, “Greater”, “Petit”)	40
Specific national, regional, or body-of-water adjectives and nouns	“British”, “European”, “Caspian” “Scandinavia”, “Atlantic”	60
Honorific titles. Here used as part of a hospital or institutional name.	“King”, “Colonel”, “Saint”	60
Common religious nouns, including proper names	“Order”, “Mercy”, “Maria”, possessives like “Paul’s”	60
Other Adjectives applicable to countries, towns, or hospitals, such as denominational terms	“Novo”, “Royal”, “Adventist”, “United”	50
Organization descriptors	“Dept.”,“College”, “Hospital”, “Center”, “Universidade”	40
Biomedical fields of study and diseases	Nouns, e.g., “Surgery”, “Pediatrics”, “Genetics”	70
	Suffixes like “…ology” in several languages	5
	Adjectives, e.g., “Health”, “Prenatal”, “Cellular”, “Microbiol”	80